

**PATENT****IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

In re Application of: Freedman  
Serial Number: 09/887,655  
Filing Date: June 22, 2001  
For: **Automated control of outbound transit  
links in a multi-homed BGP routing  
environment**

**DECLARATION OF AVI FREEDMAN**

I, the undersigned, Avraham Freedman, declare and state that:

1. I am the named inventor of the subject matter of the above-identified application. At all relevant times herein, I was Chief Network Architect for Akamai Technologies, Inc., the assignee of this application.
2. The subject matter of the invention was conceived and reduced to practice in this country well before May 29, 2001.
3. In particular, I have attached hereto as Tab 1 an email dated November 22, 2000 that I authored. In particular, this email includes several sections that refer to the "Router Buddy," which is the shorthand name for the subject matter that was later described in my patent application. The basic operation of the product was described in the TECH NOTES portion of the document for the "Router Buddy (Stage 1 Product)." Another earlier portion of the same email, titled SYSTEM COMPONENTS: ROUTER BUDDY, described the various components that had been built. They included components 2a and 2c, which were referred to in the email as the Fixer-Route Injector and the Measurer; these were the components that were used to initiate periodic path quality measurements for each of a set of transit network/destination network links, e.g., by inserting an overriding test route into the router. Component 3, the Decision System, was the component that evaluated whether a given link was a candidate for rerouting, and the Policy Adjuster, Component 1, was used to facilitate the reroute based on the


measured data. As indicated in my email, all of these components were working as of the date I wrote this email, namely, November 22, 2000. Indeed, by this date Akamai had tested the Router Buddy concept in the United States and we had established that it worked for its intended purpose.

4. Akamai employees working under my direction worked on the Router Buddy project throughout the last several months of 2000 and the Spring of 2001. Tab B is a User Manual for the Router Buddy, which was written to support the project. The User Manual is also dated November 22, 2000. Tab C is a data set that was developed based on a Router Buddy test on December 1, 2000. Tab D is a Cisco router configuration for use in implementing the Router Buddy with a Cisco router. This document is dated January 22, 2001. Tab E is a similar document that was developed for the Juniper router. It is dated January 31, 2001.

5. The Router Buddy product embodied the invention that is set forth in my patent application. The product was conceived, designed, built and tested successfully before May 29, 2001, and all such activities took place within the United States.

I further declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the above-referenced application or any patent issuing thereon.

By:



Avraham Freedman

TAB A

**Judson, David**

---

**Subject:** FW: Arriva!

-----Original Message-----

From: Avi Freedman [mailto:avi@freedman.net]  
Sent: Wednesday, November 22, 2000 10:24 AM  
To: AAhola@akamai.com  
Cc: ringel@akamai.com; strumpen@akamai.com; avi@akamai.com  
Subject: Re: Arriva!

Please read what I did below with Simon Lee last night.

Comments from Matt and Volker welcome, of course!

Avi

-----  
-----  
-----  
By 1/31/01, Akamai must deliver specification for the forwarding table for the Edge Router product.

By 6/30/01, Akamai must be able to have a productized firstpoint release that does local measurements using the standard firstpoint agent, but takes the custom per-site firstpoint map developed and exports it using the 1/31/01-due specification to the ER at the customer site.

By 9/30/01, Akamai must make a single firstpoint agent/measurement box be able to do measurements for multiple transit links.

-----  
-----  
-----  
ER:

1 box for forwarding path injection/UI  
1 box as the forwarder  
1 firstpoint probe box per transit connection

RB:

1 box for router buddy measurements/policy injection

Then adding redundancy as a feature to sell, you'd need to double boxes for each app.

-----  
-----  
-----  
SYSTEM COMPONENTS: ROUTER BUDDY

Component 1: Policy Adjuster

This component talks to Ciscos and Junipers and adjusts outbound routing policy to do the router-syntax equivalent of "send PSI traffic outbound through Sprint instead of UUNET" as well as "send traffic to keynote agent X through ATT".

STATUS: Works with Cisco, Juniper due 12/7  
STATUS: Developed by Akamai, handoff ???

## Component 2: Measurement System

### Component 2a: Fixer-Route Injector

This component injects temporary fixer-routes to the local router(s) to cause all outbound packets to the points being temporarily measured to go out a particular transit connection.

STATUS: Works with Cisco, Juniper

STATUS: Developed by Akamai

FUTURE FEATURES: Inject via local bgp feed for faster cycle times

### Component 2b: History System

Keep short history of measurements to allow "smoothing" of measurements so traffic is only adjusted if longer-lasting problems are detected.

STATUS: Not developed

### Component 2c: Measurer

Performs PING, TCP-PING, and HTTP measurements from the router or a local box to remote sites/agents, optionally causing measurement to be symmetric with a given transit provider.

STATUS: PING works with Cisco and Juniper from the router

STATUS: PING from a local box due 12/1

STATUS: Developed by Akamai

## Component 3: Decision System

Takes current and historic measurement data, combines with optional cost matrix, and decides what router policy changes should be made and how often. Also, decorrelates universally poorly performing remote data points. Also, adds in data about bad AS-AS mappings from the Akamai akanote feed.

STATUS: Working now, more knobs and tweaks due by 1/1

STATUS: Needs to integrate akanote data from Akamai

STATUS: Developed by Akamai

## Component 4: Akamai Measurement Points

Akamai will deploy 30+ machines across the Internet to be used by router buddy for measurement. Thus, instead of only pinging routers, more deterministic tests between RBs and these measurement points can be made.

STATUS: Due 2/1

STATUS: Will be done by Akamai

## Component 5: Netflow

RB box will be able to take flow stats from local router(s) and determine which ASs are the biggest 30, and test to those ASs instead of to the "global 30 big ASs" determined by Akamai.

STATUS: 2Q01

STATUS: Will be done by Arriva!

## Component 6: Install tool

STATUS: 2Q01

STATUS: Will be done by Arriva!

## Component 7: Monitoring

Monitoring of performance measurement system, router policy adjustment system,

reachability of network, etc... Done locally and form Arriva's NOC locations.

STATUS: 1Q01

STATUS: Will be done by Arriva!

Component 8: Akamai Akanote Feed

A feed of AS-AS performance data from Akamai, delivered via scp or smtp.

STATUS: 1Q01

STATUS: Will be done by Akamai

Component 9: AS and Core point Feed

A feed of top-30 ASs and core points for the top 30-200 ASs. Delivered via scp or smtp once/day.

STATUS: 1Q01

STATUS: Will be done by Akamai

Component 10: GUI

Allows user to set "knobs", prefixes and ASs and transit sessions to monitor. Shows results of current and past performance data. Shows changes being made to routers.

STATUS: Currently in beta, shows current performance measurements and router policy changes

STATUS: Ongoing

STATUS: Being done by Akamai now

-----  
-----  
-----

#### SYSTEM COMPONENTS: EDGE ROUTER

Component 1: Forwarding Engine #1

A unix-based software package that runs on commodity PC hardware to do Nx100mb or Nx1000mb forwarding of packets according to a forwarding table injected by an Arriva! module. Forwarding engine is NOT a BGP/OSPF/RIP/physical-OC3/T3/T1 router. Forwarding engine has basic user interface to query current forwarding, but expect most interaction with the UI module on the forwarding table injector.

STATUS: Not developed yet

STATUS: Not developed by Akamai

Component 2: Forwarding Table Injector

Takes the Firstpoint mapping feed from Akamai's global mapping system and injects that table (and later on the deltas for that table) to the one or more forwarding engines locally.

STATUS: Not developed yet

STATUS: Not developed by Akamai

Component 3: Juniper (for lack of better name)

Module to let the FTI inject into the Juniper forwarding backplane.

STATUS: Not developed yet

STATUS: Not developed by Akamai

Component 4: UI

Shows current state of Firstpoint feed and committed FTI updates into local router(s).  
Also, minimal GUI, but expect users will demand command line interface.

STATUS: Not developed yet  
STATUS: Not developed by Akamai

-----  
-----  
-----  
  
TECH NOTES

-----  
----- Router Buddy (Stage 1 Product) -----  
-----

WHAT IT DOES:

RB has two goals: Stability and performance.  
How measured: Keynote and other measurement agents.

Basic idea:

A network has a POP with 3 transit providers (T1, T2, and T3, say UUNET, Sprint, and ATT).

Stability:

Every 5 minutes:

- 1) Measure performance of each transit provider as seen from the local router, testing just from that same provider. So, measure T1's performance from the link to T1 only. (for symmetry)
- 2) If any one provider is having internal reachability troubles, stop using it. (shut down the session and soft clear)
- 3) If > 1 provider is having internal reachability troubles, sound highest level alarm and unshut any shut session.

Then, performance more and stability less:

Take the set of big networks and sort them by size of traffic, based on Akamai's measurements. Call these AS1 ... AS30.  
Chances are, T1, T2, and T3 will also overlap with AS1-30.

(Not going higher than 30 because we want to finish testing so we can iterate frequently)

So, for performance:

- 1) Measure performance of each remote AS1-30 as seen from each local transit provider.
- 2) Each round of running, take the biggest AS that we have problems to, and move traffic to that AS to a better-performing connection (selected randomly for now).

- ignoring latency for now, just look at ping loss

Also, for people who want it:

- 1) Take a list of IP prefixes to test to (a la Keynote agents)
- 2) Test performance to each IP from each transit connection
- 3) Reroute as needed for optimal loss and latency

Architecture:

- 1) A local tester to test quality of remote ASs as seen from the local router, through each of T1, T2, ... in turn
- 2) A "whisperer" to tell the router the new AS-AS preference rules
- 3) Hooks to Akamai to give info about:

- a) AS-AS performance from Akanote
- b) List of ASs, ranked by size
- c) List of interesting points in each AS, and measurement method
- d) Right to test to Akamai boxes in those ASs for better measurements

How do we measure an AS?

- 1) First, we connect to the router and set it so that the outbound and inbound paths to and from the points being measured go first from T1, then T2, then T3.
- 2) For a given transit provider's measurements, we take ping measurements from the router to the "core points" specified for each AS. Right now, this is static. Soon, the list becomes dynamic and comes from Akamai. Soon, we add HTTP measurements to Akamai boxes and also the ability to do round-trip and one-way latency and loss tests.

How do we do the move?

- 1) We tell the router in config format the equivalent of "Please move all traffic to PSI to go through our Sprint connection instead of our UUNET connection". This is done via route-maps and as-path access-lists on Ciscos, and soon on Junipers.

How do we select what needs to move?

- 1) Currently, do the "biggest AS that's having a problem".
- 2) Soon, keep history and only move an AS if it's been poorly performing for N rounds.
- 3) Soon, have a cost matrix to say "when moving an AS, prefer T1 over T2 and T3 if all else is equal". Useful if T2 and T3 are smaller or more expensive.
- 4) Soon, a feed from Akamai of AS-to-AS performance will be used to find other "bad paths" that local measurement cannot detect.

How is it going to be made better/faster?

- 1) Measurements will be made faster by:
  - a) Doing ping-testing from the router-buddy box, instead of from the router (can decrease from 5-20 minutes to 10 seconds)
  - b) Injecting "fixer routes" to control routing to the test points using a BGP process running on the router-buddy box (can cut off 20 seconds from each round).

Control of routing to prefixes.

- 1) Soon, a system to test connectivity to a "Class C"-sized block and inject a re-routing statement to the router based on best performance, will be implemented.

Also to be done:

- 1) Documentation
- 2) A GUI for control and viewing of data
- 3) An alert system for paging and e-mail escalating alerts
- 4) Presenting longer-term AS-to-AS performance data to measure each transit provider (like Keynote)
- 5) System to take "flows" from local routers to dynamically configure which the big/interesting ASs are

Non-product to be done:

- 1) Set up a lab of 20 Tls with routers and boxes to use to test performance of router-buddy vs. inap, etc...



-----  
----- Edge Router (Stage 2 Product) -----  
-----

The goal of Edge Router is to control outbound packet flow in a much more fine-grained way than router-buddy.

Router-buddy says "Take all 3,000 PSI prefixes and prefer them through Sprint instead of UUNET" by saying "match \_174\_ in the AS\_PATH of the routers heard from 1239 and set the LOCAL\_PREF to 1000000".

Edge Router knows about the best way to get to all 3,000 of PSI's prefixes, which will not ALL be through one of Sprint or UUNET.

To get that map, a local firstpoint probe is set up. That probe does local tests suggested by the global firstpoint system. The global firstpoint system then feeds the map back into the edge router local firstpoint probe box.

That firstpoint local probe box then either feeds the 400,000+ routes to the local router (if it's a Juniper or someone we've worked with), or to the Arriva!-built forwarding device.

Tasks:

- 1) How to feed the routes to the local router
- 2) Synchronization for feeding routes to two local routers
- 3) Akamai: Send the prefix table down to the local box

-----  
----- Firstpoint -----  
-----

As a reminder, Arriva! has a right to sell Firstpoint as-is, to be used for load-balancing inbound and/or outbound applications for the multi-homed.

It can only balance outbound from Web servers or NATs, not general IP packets coming into the router...

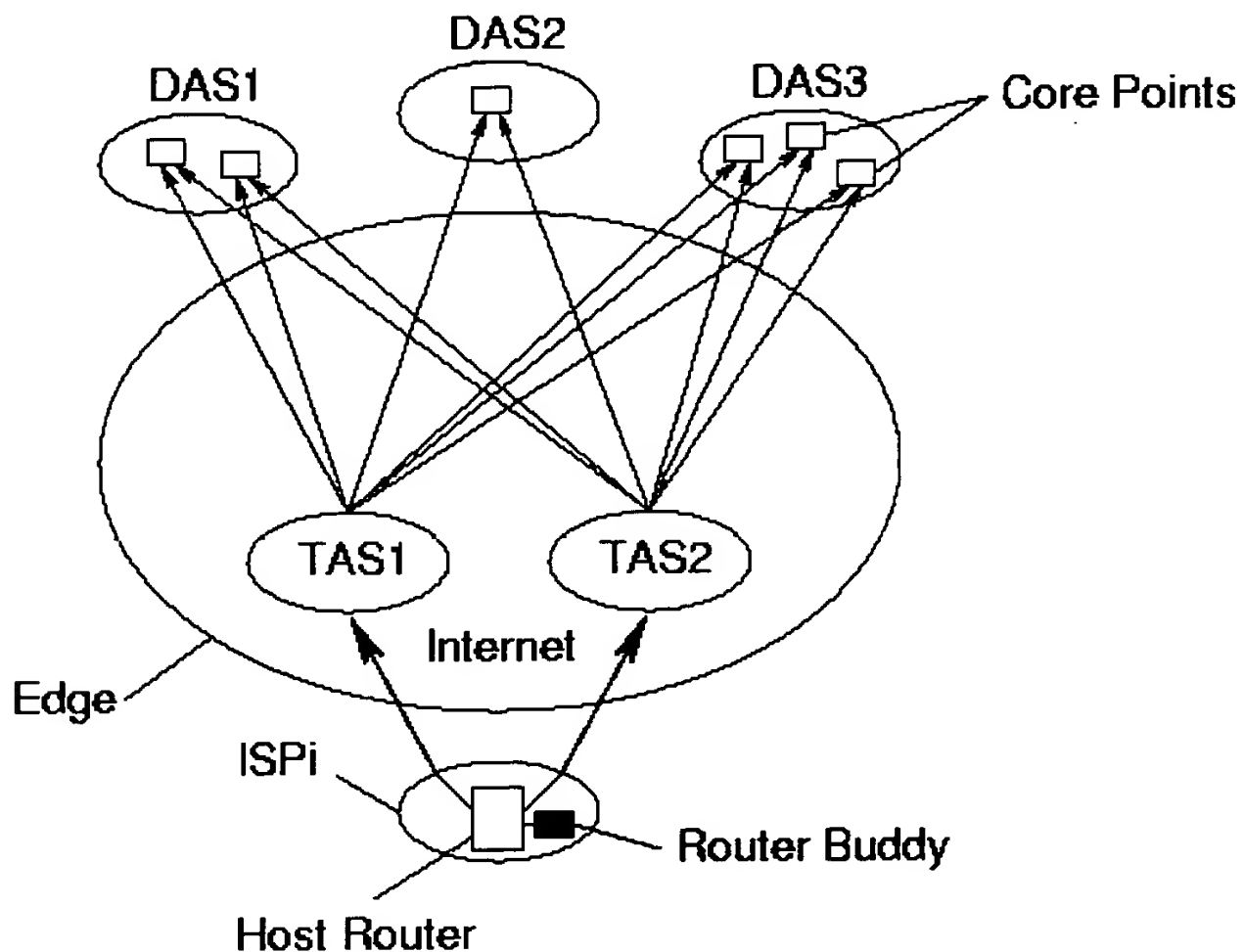
It does that balancing by controlling the inbound path, and setting up routing so that if mapped to IP X on the inbound, IP X belonging to the Sprint connection, then all outbound goes to sprint. The user has to set the outbound routing to do that using router rules or physical wiring.

Edge router is needed to do that.

TAB B

## Router Buddy User Manual

Router Buddy provides automatic routing configuration for multihomed routers based on global traffic analysis of the Internet. Router Buddy operates in the environment shown in Figure 1.



**Figure 1.** Router Buddy environment

Router Buddy is connected to a multihomed **host router** at the edge of the Internet, for example **ISPi**, an ISP with router buddy turbo injection. The multihomed host router in Figure 1 is connected to the Internet via two **transit AS's**, namely **TAS1** and **TAS2**. Any traffic from the host router to a **destination AS** flows through one of the transit AS's. Figure 1 shows three potential destination AS's **DAS1**, **DAS2**, and **DAS3**. Router buddy conducts local traffic analysis of outgoing packets to a set of machines, called **core points**, representative for the destination AS.

## Transit Link Selection

The algorithm for transit link selection uses a 2-dimensional cost matrix. Each row corresponds to a destination AS, and each column corresponds to a transit AS. The matrix elements record the sum of the packets lost for all core points of a destination AS, when routing packets through the associated transit AS. Below is an example for a cost matrix based on packet loss information:

Loss	TAS1	TAS2
DAS1	2	9
DAS2	0	3

<b>DAS3</b>	6	14
-------------	---	----

The first step of the link selection algorithm is to transform the loss matrix into a **bit matrix**. Each matrix element of the bit matrix stores one bit of information, which is whether the link to the associated destination AS via the associated transit AS is considered good (value 1) or bad (value 0). The transformation from loss to bit is a simple configurable threshold computation. For the example that leads to the bit matrix below, we assume that a link is considered bad if more than 75% of the packets are lost.

Bit	TAS1	TAS2
<b>DAS1</b>	1	0
<b>DAS2</b>	1	0
<b>DAS3</b>	1	0

The second step of the link selection algorithm computes the **column sum** of the bit matrix:

TAS1	TAS2
3	0

The third step of the link selection algorithm is to **pick** a particular transit AS. The pick is based on the transit link quality represented by the column sum, a configurable link preference, and a random tie breaker. For the column sum above, the clear winner is transit AS **TAS1**, because it has the maximum link quality value 3 (as opposed to 0).

After each round of measurements, the link selection algorithm executes to pick a transit AS for **one** destination AS.

## Demo Setup for Cisco's

1. Install user, password, and enable password on host router
2. Inspect host router configuration for **next-hop IP's** and **available access-list identifiers** (2 per transit AS)
3. Configure transit AS's (file config/tras)
4. Supply authentication for host router with `interd -n -r`
5. Check authentication `interd -a -n -r`
6. Demo run: `interd -a -n`
7. If you ran `interd` without the `-n` flag, cleanup the router with `interd -a -r`

---

\$Id: manual.html,v 1.3 2000/11/22 15:09:23 strumpen Exp \$

TAB C

# Router Buddy Measurements

Router: 192.41.177.87  
Date: Fri Dec 1 19:08:41 UTC 2000  
Ping ratio: 75%

	Link Quality	Transit AS									
		174	701	1239	2548	2685	2914	3491	5000	6461	7960
Desti- nation AS	31	50	88	100	100	88	100	88	100	100	75
	174	95	100	100	98	98	92	98	95	98	72
	702	0	0	0	0	0	0	0	0	100	0
	1239	67	67	62	54	62	54	67	67	67	0
	1740	81	94	94	94	100	100	100	100	100	0
	1791	94	88	100	100	94	94	100	100	100	0
	2914	78	97	97	97	100	97	97	94	100	84
	3356	75	81	79	81	75	79	83	77	83	71
	3967	95	93	91	96	98	98	96	96	100	82
	4713	38	50	50	50	50	50	50	44	50	44
	6461	88	88	88	100	100	94	100	100	100	88
	10593	62	58	67	67	67	62	62	62	67	54
	11486	100	100	100	100	100	88	100	100	100	94
	164.124.101.0/24	100	88	94	100	100	100	100	100	100	94
VIPath	203.248.240.0/24	0	0	0	0	0	0	0	0	0	0
	203.8.183.1/32	88	100	100	100	100	100	100	100	100	88

This page is generated automatically -- your feedback is encouraged.

TAB D

## Cisco Configuration

### Outbound Routing at AS Granularity

We route packets for a particular **destination AS** through a particular transit AS. (In the future, we may consider a finer granularity by routing CIDR blocks instead.) The transit AS is selected by means of a route map which sets a local preference value high enough to enforce selection of the transit AS. Here is an example of a router configuration generated automatically by router buddy:

```
router bgp 64515
  neighbor 207.106.2.74 remote-as 6461
  neighbor 207.106.2.74 soft-reconfiguration inbound
  neighbor 207.106.2.74 route-map AKATRANSIT_6461 in

ip as-path access-list 178 permit _174_
ip as-path access-list 178 permit _3356_
ip as-path access-list 178 permit _4713_

route-map AKATRANSIT_6461 permit 10
  match as-path 178
  set local-preference 5000

route-map AKATRANSIT_6461 permit 1000
```

Initially, the as-path access-list denies all AS's.

```
router bgp 64515
  neighbor 192.41.177.80 route-map AKATRANSIT_5000 in

ip as-path access-list 175 deny .*

route-map AKATRANSIT_5000 permit 10
  match as-path 175
  set local-preference 5000

route-map AKATRANSIT_5000 permit 1000
```

When switching the transit link for a destination AS, we swap access-lists. In the following, I describe which commands router buddy executes on a Cisco router. As a running example, assume that we want to route traffic to destination AS 702 through transit AS 5000. For the sake of brevity, I show the configuration commands and associated configuration snippets for the case that the router has just been initialized.

#### 1. Command `show route-map AKATRANSIT_5000` yields:

```
route-map AKATRANSIT_5000, permit, sequence 10
  Match clauses:
    as-path (as-path filter): 175
  Set clauses:
    local-preference 5000
  Policy routing matches: 0 packets, 0 bytes
route-map AKATRANSIT_5000, permit, sequence 1000
  Match clauses:
  Set clauses:
  Policy routing matches: 0 packets, 0 bytes
```



This configuration snippet shows the initial route map configuration, that permits all routes through the route map with sequence number 1000. This route map is in place to permit routes without assigning the local preference in case that the access list associated with the route map of sequence number 10 denies all routes. If router buddy had assigned destination AS 702 to another transit AS already, we would inquire about the route map of the current transit AS in addition to the new transit AS 5000. Router buddy maintains the state of the assignment transit assignment, knowing which routemap to retrieve from the host router.

We compare the access-list number for the as-path filter (175 in the example above) with the expected ones. (The administrator must specify two access-list numbers in configuration file tras!) If none of the expected numbers is found, we signal an error. Otherwise, we use the retrieved access list number 175 to inquire about the currently installed access list.

2. Command `show ip as-path 175` yields:

```
AS path access list 175
  deny .*
```

If the access list stanza contains a "permit" or "deny," we continue processing. In this example, no route will pass through access list 175. This configuration corresponds to the initial state set by router buddy.

If the router were not in the initial state, the access list would permit a list of destination AS's. In addition to retrieving the access list of the new transit AS 5000 for destination AS 702, router buddy would also retrieve the access list of the current transit AS.

3. In the current example, router buddy proceeds by modifying the access list stanza to permit destination AS 702, and subsequently installing this stanza under a new access list (176). Then, the access list is swapped in the route map from 175 to 176. This procedure ensures that at any point in time a valid access list is referenced by the route map; an important issue, because Cisco routers reportedly crash otherwise. The stanza the `deny .*` causes stanza `permit _702_` to be installed for access list 176, leading to the following configuration for transit AS 5000:

```
ip as-path access-list 176 permit _702_

route-map AKATRANSIT_5000 permit 10
  match as-path 176
  set local-preference 5000

route-map AKATRANSIT_5000 permit 1000
```

The following commands are executed by router buddy to obtain this configuration:

```
ip as-path access-list 176 permit _702_      # set new acl
route-map AKATRANSIT_5000 permit 10          # set new route map
no match as-path 175
match as-path 176
no ip as-path access-list 175                # delete old acl
```

If the destination AS is assigned to a transit AS already, router buddy deletes the destination AS from the access list stanza of the current transit AS. If the destination AS is the only one left, change the stanza from `permit <destAS>` to `deny .*`.

4. Router buddy filters all routes received from transit AS 5000 by executing command:

```
clear ip bgp <next hop IP> soft in
```

If the destination AS is deleted from another transit AS to transit AS 5000, the routes of that transit AS must be processed as well.

## VIP Routing

Very Important Paths are configured with simple static routes for CIDR blocks:

```
ip route 207.106.125.0 255.255.255.0 207.106.10.26
```

## Source-based Policy Routing (Superping)

**Important:** source-based policy routing may not work on every router. For example, Cisco's model 12000 (GSR) does not support policy routing at all.

Here is a sample configuration of a Cisco router using source-based policy routing to forward all icmp echo packets from router buddy 10.18.110.2 through the link connected to next-hop 10.18.109.130:

```
interface FastEthernet0/0
ip address 10.18.110.1 255.255.255.0
ip route-cache policy
ip policy route-map AKASUPERPING

access-list 110 permit icmp host 10.18.110.2 any echo

route-map AKASUPERPING permit 10
match ip address 110
set ip next-hop 10.18.109.130
```

### Comments:

1. The following parameters of router buddy must be configured when using superping with source-based policy routing:
  1. Router interface connected to router buddy.
  2. IP address for router buddy or as many IP addresses as there are transit AS's.
  3. If only one IP address is available rather than as many as there are transit AS's for use by router buddy, specifying exactly one IP implies that the next-hop clause in the route-map will have to be reassigned between superpings.
  4. An otherwise unused access list identifier in range [100 - 199], or as many as there are transit AS's.
  5. Verify that route-map name **AKASUPERPING** is otherwise unused.
2. We use an extended access list to limit the source based policy routing to icmp echo packets. Note that Cisco's extended access lists require an integer identifier in the range [100 - 199].
3. When using multiple IP's for interface aliases on router buddy, one per transit AS, use multiple sequence numbers for the same route map. For example:

```
access-list 110 permit icmp host 10.18.110.2 any echo
access-list 111 permit icmp host 10.18.110.3 any echo
access-list 112 permit icmp host 10.18.110.4 any echo
```

```
route-map AKASUPERPING permit 10
  match ip address 110
  set ip next-hop 10.18.109.130
```

```
route-map AKASUPERPING permit 20
  match ip address 111
  set ip next-hop 10.18.109.134
```

```
route-map AKASUPERPING permit 30
  match ip address 112
  set ip next-hop 10.18.111.130
```

## Router-Cluster: Static Route Redistribution

On the router with next-hop IP 2.2.2.2, use the following configuration to redistribute the static route for 1.1.1.0/24 to all other routers speaking iBGP:

```
ip route 1.1.1.0 255.255.255.0 2.2.2.2
access-list 10 permit 1.1.1.0 0.0.0.255
route-map AKAVIPREDIST permit 10
  match ip address 10
router bgp 1
  redistribute static routemap AKAVIPREDIST
```

---

\$Id: cisco.html,v 1.8 2001/01/22 21:14:11 strumpen Exp \$

TAB E

## Juniper Configuration

### Outbound Routing at AS Granularity

We route packets for a particular **destination AS** through a particular transit AS. (In the future, we may consider a finer granularity by routing CIDR blocks instead.) The transit AS is selected using a policy-statement that accepts routes by mean of AS path selection and attaches a high local-preference value to enforce the preferred next-hop transit link. Here is an example of a router configuration generated automatically by router buddy:

```
policy-options {
  policy-statement AKATRANSIT_6172 {
    term first {
      from {
        protocol bgp;
        as-path AKAPATH_6172;
      }
      then {
        set local-preference 5000;
        accept;
      }
    }
  }
  as-path AKAPATH_6172 ".* (705) .*";
}
```

Initially, we install the policy statement and an as-path access-list that *does not match any route*. Since JUNOS does not support a logical not for as-path regular expressions, we match for a highly unlikely (and illegal) path consisting of 10 consecutive 0's:

```
as-path AKAPATH_6172 0{10};
```

When switching a destination AS from the current transit AS to another, new transit AS, we add the destination AS to the regular expression of the as-path access-list of the new transit AS and delete the destination AS from the regular expression of the as-path access-list of the current transit AS. The policy statement remains unchanged.

### VIP Routing

Very Important Paths are configured with simple static routes for CIDR blocks:

```
routing-options {
  static {
    route 164.124.101.0/24 nexthop ge-0/0/0.0;
    route 203.8.183.1/32 nexthop ge-0/0/0.0;
  }
}
```

### Router Cluster

In a router cluster, we use IBGP to redistribute the local preference settings for outbound routing. No special configuration is required other than running IBGP.

VIP routes require redistribution of static routes. JUNOS readvertises static routes by default. Thus, if an IBGP protocol is configured appropriately, static routes set on one router will automatically propagate throughout the router cluster.

Note that readvertisement of static routes by default is dangerous, since it may cause undesirable effects such as routing loops. Static routes other than router buddy's can be prevented from readvertisement by appending the `no-readvertise` option to the static route statement; for example:

```
routing-options {
  static {
    route 1.1.1.0/24 nexthop ge-0/0/0.0 no-readvertise;
  }
}
```

## Administrational / Installation Procedures

1. Install router buddy's **user key** on Juniper. For example:

```
system {
  login {
    user netarch {
      uid 2001;
      class superuser;
      authentication {
        ssh-rsa "1024 33 <public key> netarch@localhost.localdomai
      }
    }
  }
}
```

2. Install an **import policy** in the BGP group associated with each of the relevant transit AS's. Use the policy name: `AKATRANSIT_<peer-as>`. Here is an example of adding an import statement, **if no import statement existed beforehand**:

```
protocols {
  bgp {
    group ATHOME {
      passive;
      import AKATRANSIT_6172;
      export outbound;
      peer-as 6172;
      neighbor 209.219.187.5;
    }
  }
}
```

**If your BGP group has an import statement already**, we suggest to use a sequential list: If `<policy expression>` matches and the action is accept, policy `AKATRANSIT_6172` is evaluated:

```
import <policy expression>
```

becomes:

```
import [<policy expression> && AKATRANSIT_6172]
```

**IMPORTANT:** If `<policy expression>` rejects routes that contain the destination AS's in their path, router buddy will be ineffective! Consult the administrator to discuss this situation.

3. In a router cluster, run IBGP to distribute aspath local preferences and to redistribute static routes.
4. Ensure that all transit links have a full EBGp feed, i.e. import routes to all destination AS's.
5. When configuring VIP networks, be sure that these networks do not overlap with CIDR blocks owned by any of the transit AS's that are also destination AS's. This may cause routing loops! (TODO: Investigate!)

---

\$Id: juniper.html,v 1.6 2001/01/31 15:30:57 strumpen Exp \$